

Building A Diskless Folding @ Home Farm

Project: [Folding @ Home](#)

Date Written: November 25, 2002

Written By: [Jason Rabel](#)

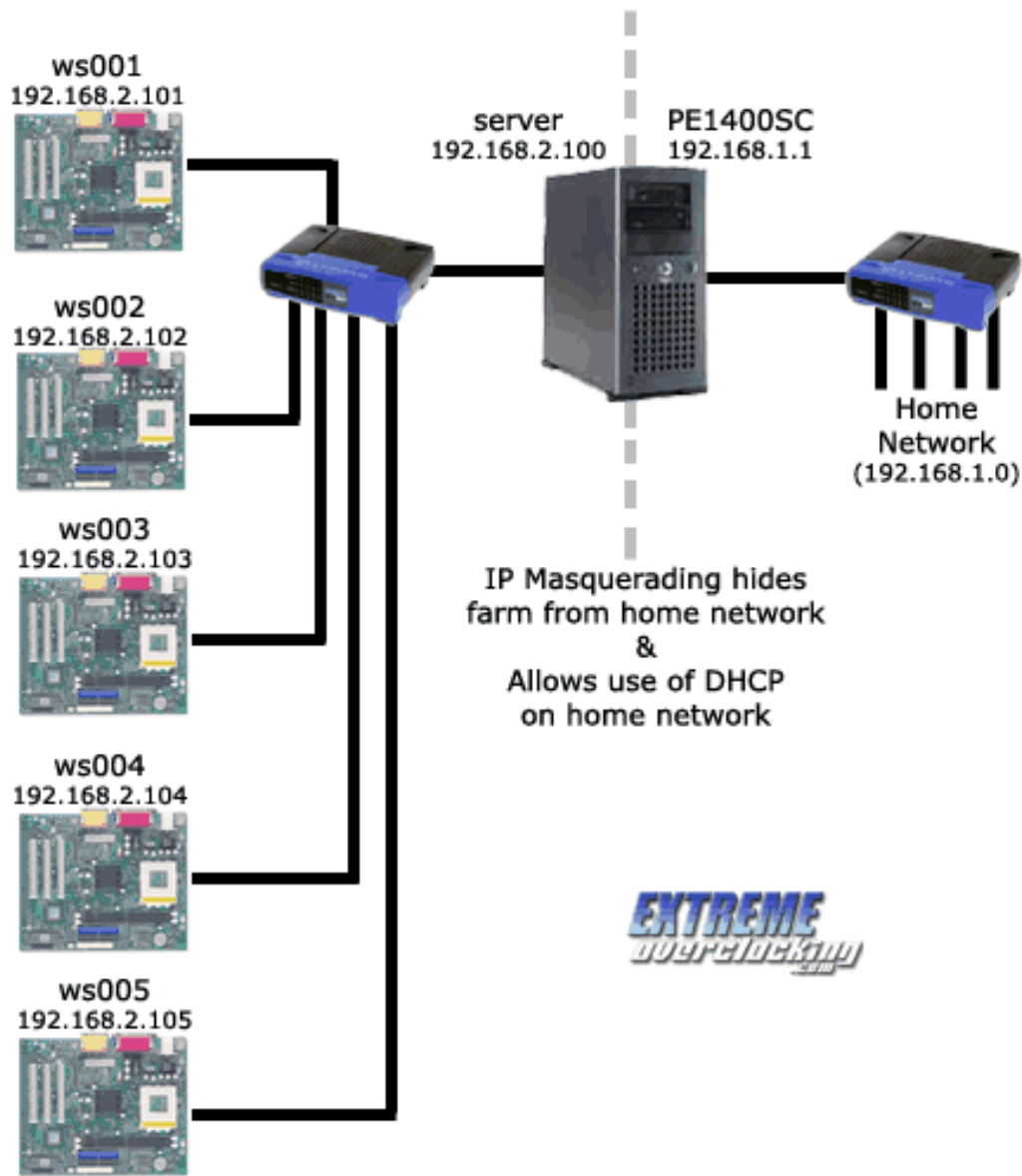
Introduction:

No doubt you have heard of the [Folding @ Home](#) project, which has been going on for well over a year now. The theory is the client uses your computer's idle CPU cycles to fold proteins to help find medical cures. There are statistics for each individual & team (if you decide to join one) showing how much work has been done, and overall rank. This healthy bit of competition has led people to dig up and recycle whatever working piece of PC equipment they can find to "fold".

As a folding farm starts to grow, just like in a corporate network you start to experience growing pains and management of all the machines can become a real nightmare. Each machine running its own OS and just folding seems simple enough, which it can be, however the model can be simplified down even more. When I was making folding machine I would try to dig up any hard drive I could that was about 500MB or bigger for whatever bargain price someone was willing to accept. Usually they would run fine for a few weeks then die, or just act up randomly. Needless to say this was a real pain.

By setting up folding machines to boot over the network, it eliminates the need for a hard drive, and even a video card. Not only does this save you \$\$\$ money \$\$\$ by not needing as much hardware (which is always good), but it also reduces the amount of power consumed (another bonus), and reduces the amount of heat generated (because you are consuming less power). Another BIG reason to boot from a common file system like this is that it greatly reduces the effort required to manage & update the systems.

Looking at the picture below, in a nutshell, this is what I have done at home. I have created a separate network for my folding farm, which is tied into my existing network so the clients have internet access (to send & receive work).



***Side Note:** At first I had everything on one subnet, but that meant disabling the broadband router's DHCP server & setting my regular work machines with static IPs. That had to be done because you can only have one DHCP server on the network, otherwise problems can occur. This was a real pain since my network is ever changing, plus I wanted my farm to be a little less obtrusive. So instead I decided to create a separate network just for the farm, and the folding server uses IP Masquerading to route packets between the two networks, much like your broadband router routes packets between your home network and the Internet. The only downside is you need a separate hub / switch for the farm, however I think a cheap 10Mb hub would be more than adequate.

This guide was written with a lot of help from a few other sources, with two big ones being key in my success. The first of course being the [LTSP documentation](#), which is also available in [other languages](#). The second source was from a [diskless boot setup for a Prime & SETI "monster" farms](#). My main motivation for writing this guide is to clear up some of the parts that the other ones were a bit fuzzy in, or completely omitted information. If you feel that a certain part of this guide is a little light on information, I suggest checking those links for more info first. Also some parts in this guide I tried to provide links to other sites that had specific info that would be helpful, like the [IP Masquerading site](#).

What is needed:

To start out, you will need the following:

- A PC with a hard drive & two NICs - This will be the "Farm Server"
- At least one PC with a single NIC and a video card in it (for testing, afterwards you can remove the video card) - These will be the "Farm Clients"
- A switch or hub, and enough cable to hook up everything
- A copy of your favorite Linux distro (that is on the [LTSP supported list](#)), some LTSP files, and the FAH client (more details on these later)

Getting Started:

When setting up my farm, I came across having to make a decision to either have each client machine make a RAMdisk big enough to hold the F@H files and fold from there, or to each use their own NFS mounted directory and work over the network. Changing the size of the RAMdisk from the kernel's default meant having to compile a custom kernel, which I really didn't want to do, also if a system lost power, all the work in progress would be lost. Running over the network really isn't as bad as it sounds. The F@H client only writes data when it finishes a frame, and it is a very small amount of data at that. Also if a client loses power or locks up, the data is safely stored on the server and it can resume where it left off. Also having all the files in one central place makes it much easier to maintain and monitor. Since the farm is also on its own physical network, it shouldn't interfere with your local network at all except when it is having to send / receive WUs.

My setup that I will be going through in this article revolves around Red Hat 8.0. [LTSP](#) actually supports 6 different Linux Distros, so if you feel more comfortable with a different one, you can use it just as easily. The configuration files used in this setup are pretty common across all distros. Also, to save space I'm only going to list the first couple client machines in configurations, adding more clients should be as simple as copy & paste then incrementing the numbers.

If you plan on building your farm out of what you have just laying around, then I would pick the most powerful machine to be the Farm Server (it can still do F@H so don't worry), or if you are planning on purchasing identical hardware for your farm you can use one of those machines just as easily, all you need is a decent sized hard drive (5-10GB should be more than enough), an extra NIC (to connect your two networks), and I would spring the money for a little extra RAM (256MB should be plenty).

The first thing I did was to do a pretty bare install of Red Hat on the Farm Server. The install process is pretty basic, if you need help getting it installed there are plenty of guides on the net to show you how. The only difference I did from the standard install is that I manually configured my NICs (usually they default to DHCP). I configured both NICs with static IPs, the first NIC was an IP within my current home network, and the second NIC was for the farm network and had the IP 192.168.2.100. Refer to the picture in the introduction for how my IPs are setup.

Next I made sure I had the following services installed:

- bind
- dhcp
- nfs-utils
- tftp-server
- portmap

To check and see if they are installed, you can issue the following commands, and they should return the version that is installed:

```
[jason@PE1400SC root]$ rpm -q bind
bind-9.2.1-9
[jason@PE1400SC root]$ rpm -q dhcp
dhcp-3.0pl1-9
[jason@PE1400SC root]$ rpm -q nfs-utils
nfs-utils-1.0.1-2
[jason@PE1400SC root]$ rpm -q tftp-server
tftp-server-0.29-3
[jason@PE1400SC root]$ rpm -q portmap
portmap-4.0-46
```

If you are missing one or more packages, you can use the Red Hat up2date program, or you can find and install them from Rpmfind.net. Be sure to pick the proper files for your distro.

Once you have verified that you have the necessary services installed (but not started), you can go ahead and install the LTSP RPM files. I have a mixture of NICs that I use in my farm, including some Intel NICs, and onboard NICs which boot via PXE (slightly different than the regular DHCP process), I installed those files too, which the GZip includes a README on where to put the files and the extra configuration necessary.

[The Download Page @ SourceForge](#) has all of the following files, and much more should you need them.

***Note** - There might be slightly newer releases depending on when you read this article, so the version numbers might not match up 100%. Download the files to a directory on the Farm Server then install them via the regular RPM install process. Also, I don't know if you can just do a *.rpm to install them, I didn't try that, instead I installed the files in that order (there are only 3 of them so it's not that big of a deal), and uncompressed the extra PXE file. One of the files says "local apps" in it, but you aren't going to be running anything locally really, there are files that it has for DNS resolution that the clients will need, I had that hardest time with the clients resolving DNS names until I found a random post while searching the net which said to try and install that RPM, and sure enough things started working after that. You can fully configure the local apps part if you want to be able to telnet to the clients and such, but for this folding project it really isn't needed.

- ltsp_core-3.0.7-0.i386.rpm
- ltsp_kernel-3.0.5-0.i386.rpm
- ltsp_local_apps-3.0.0-0.i386.rpm
- pxestuff-3.0.5-i386.tgz (*Note - Only needed if you are going to use NICs that boot via PXE)

```
[root@PE1400SC root]# rpm -ivh ltsp_core-3.0.7-0.i386.rpm
[root@PE1400SC root]# rpm -ivh ltsp_kernel-3.0.5-0.i386.rpm
[root@PE1400SC root]# rpm -ivh ltsp_local_apps-3.0.0-0.i386.rpm
[root@PE1400SC root]# tar zxvf pxestuff-3.0.5-i386.tgz
```

Now you need to run the initialization program, you really don't need to change any settings that it asks.

```
[root@PE1400SC root]# cd /opt/ltsp/templates
[root@PE1400SC root]# ./ltsp_initialize
```

Just a note before we get into the custom configuration part, on the last page I will put up all the config files listed in this how-to in a compressed file for download. However I highly suggest reading through everything here completely so that you can get a better understanding of how everything works.

Now it's time to get down to the nitty gritty and configure the files. The first file is the `dhcpd.conf` file, which tells all the clients what file to load and what their network settings are. The DNS server I use is my broadband router's IP address, yours may vary. I also put in a subnet range of `.101 - .200` which gives me room for more than enough folding machines, you can adjust to bigger if you really need it (the thought of that many farms running in my house makes me think of the fire department).

Each client machine gets a hostname of "ws00x" where "x" gets incremented for each new machine, this is done mostly for simplicity, as we go into the other configuration files you will understand better why it is done like this. The main thing you will need change is under each host ws00x you will need to find out each clients MAC address to enter. The filename you won't need to adjust, but make sure the "vmlinuz-2.4.19-ltsp-1" file exists in the `/tftpboot/ltsp` directory. The PXE booting is a little different, the necessary files are in that separate GZip file mentioned before, along with instructions on where to put each file and the configuration differences (which are for the most part like below).

Here's a picture of the back of one of my NICs as an example. The MAC address of the NIC below is 00:10:B5:C0:C2:F6.



Don't worry about the farm's dhcp server interfering with your broadband router's dhcp server. Because you are only specifying the network that is on `eth1` (the folding farm), it doesn't listen for or respond to dhcp requests on `eth0` (your existing home network). This is nice because you can add on the farm without having to change anything on your existing network.

`/etc/dhcpd.conf`

```

# Make changes to this file and copy it to /etc/dhcpd.conf
#
# If the setting below doesn't work,
# then try changing to the following setting:
# ddns-update-style ad-hoc;
ddns-update-style none;

default-lease-time 21600;
max-lease-time 21600;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.2.255;
option routers 192.168.2.100;
option domain-name-servers 192.168.1.254;
# You can set the domain name below if you have one,
# otherwise it's okay to leave it commented out.
# option domain-name "localdomain.com";
option root-path "192.168.2.100:/opt/ltsp/i386";
option option-128 code 128 = string;
option option-129 code 129 = text;

shared-network WORKSTATIONS {
    subnet 192.168.2.0 netmask 255.255.255.0 {
        range 192.168.2.101 192.168.2.200;
    }
}

group {
    use-host-decl-names on;
    option log-servers 192.168.2.100;
    host ws001 {
        hardware ethernet 00:10:B5:9B:BC:65;
        fixed-address 192.168.2.101;
        filename "/lts/vmlinuz-2.4.19-ltsp-1";
    }

# This client boots via PXE,
# that is why some of the settings are different

    host ws002 {
        hardware ethernet 00:D0:B7:48:2E:20;
        fixed-address 192.168.2.102;
        filename "/lts/pxelinux.0";
        #The next line wraps with all the digits, but it should be only 1 line with no spaces
        option vendor-encapsulated-options 09:0f:80:00:0c:4e:65:74:77:6f:72:6b:
20:62:6f:6f:74:0a:07:00:50:72:6f:6d:70:74:06:01:02:08:03:80:00:00:47:04:80:00:00:00:ff;
    }
}

```

The hosts file you will need to enter the IP & hostname of the server & each client machine, otherwise you could run into errors.

/etc/hosts

```
#Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1 PE1400SC localhost.localdomain localhost
192.168.2.100 server
192.168.2.101 ws001
192.168.2.102 ws002
```

The lts.conf file is pretty basic, when you add more machines you just add a new ws00x entry. For running just folding you shouldn't need a swap file, but you can create one should you deem it necessary or you are running something else that takes up more RAM. The RCFILE_01 entry tells it to run the folding @ home script, which I will go into more detail later. I set the RUNLEVEL at 3, which will put you at the shell on the client machine so you can do some diagnostic stuff to make sure everything is working, if you set the RUNLEVEL to 4 then you will only get a prompt on the client machine which will let you telnet to the farm server. It really shouldn't matter which you set it at since they aren't going to be hooked up to anything. I leave them at 3 because if a client doesn't look like it is folding I can hook up a monitor & keyboard to try and diagnose the problem locally.

/opt/ltsp/i386/etc/lts.conf

```
#
# Config file for the Linux Terminal Server Project (www.ltsp.org)
#
[Default]
SERVER = 192.168.2.100
DNS_SERVER = 192.168.1.254
# SEARCH_DOMAIN = "your.domain.org"
USE_XFS = N
LOCAL_APPS = N
RUNLEVEL = 4
USE_NFS_SWAP = N
#-----
[ws001]
USE_NFS_SWAP = N
SWAPFILE_SIZE = 32m
RUNLEVEL = 3
RCFILE_01 = startfah

[ws002]
USE_NFS_SWAP = N
RUNLEVEL = 3
RCFILE_01 = startfah
```

The exports file contains the directories that each client will mount. The first two are your root file system, and a swap file directory (should you ever choose to use swapping). Since the F@H client requires its own directory to do work, I created a /fah directory off of the root directory on the Farm Server, then each client

gets their own directory names ws00x, the directories need to be the same as the hostname in the dhcp.conf file to be mounted correctly. When initially creating the directory, you only need the F@H executable and the config file it generates, the rest of the files will be auto-generated. Do **not** copy any of the other files as one of them is supposed to be unique for each machine!

Just to note again, I chose to store the folding directory on the server incase a client crashed, or had to be rebooted. If it was stored merely in RAM then you would loose all your previous work. Also you can monitor the client logs on the server much easier to make sure they are running.

/etc/exports

```
## LTS-begin ##
# Note, some lines might wrap because of length, each entry
# should be only one line!

/opt/ltsp/i386 192.168.2.100/255.255.255.0 (ro,no_root_squash,sync)
/var/opt/ltsp/swapfiles 192.168.2.100/255.255.255.0 (rw,no_root_squash,async)

/fah/ws001 192.168.2.100/255.255.255.0 (rw,no_root_squash,sync)
/fah/ws002 192.168.2.100/255.255.255.0 (rw,no_root_squash,sync)

## LTS-end ##
```

The rc.local file is really long, however you will only need to make a few changes to the default config file so that you are mounting the proper folding directory. **BE SURE TO MAKE A BACKUP OF THIS FILE BEFORE EDITING!** Look below for the section that says "Mount file systems for the FAH client.", that part you will need to add in, and the other mounting part gets commented out. I also set the DEFAULT_SERVER here. Once you feel comfortable with the config file you can delete out other stuff that is not needed, since you are really only using a small portion of the file. I highlighted the changed part in **red** so you could find it easier.

/opt/ltsp/i386/etc/rc.local

```
#!/bin/sh
#
# rc.local
#
# This script will setup the environment
# for a diskless workstation, as part of
# the Linux Terminal Server Project (http://www.LTSP.org)
#
PATH=/bin:$PATH; export PATH
. /etc/ltsp_functions

/sbin/devfsd /dev

#
# Mount /proc before /tmp, so that we can display the progress
#
echo "Mounting /proc filesystem"
```

```
mount -n -t proc /proc /proc

#
# Un-mount the initrd, to free up some space
#
umount /oldroot > /dev/null 2> &1

#
# Get the runlevel for this workstation
#
RUNLEVEL=`get_cfg RUNLEVEL 5`

#
# Find out if we want to allow local applications
#

pr_set 67 "Checking for Local Apps"

LOCAL_APPS=`get_cfg LOCAL_APPS N`

if [ "${LOCAL_APPS}" = "Y" ]; then
if [ ! -d /home ]; then
pr_warn
echo
echo "WARNING! You have Local apps enabled in lts.conf, but"
echo " you don't have the local_apps package loaded"
echo " Local apps will be disabled."
echo
LOCAL_APPS="N"
fi
fi

#
# Find out if we want to swap via NFS
#
USE_NFS_SWAP=`get_cfg USE_NFS_SWAP N`

#
# Create and mount the ramdisk as the /tmp filesystem This
# is the only place we can write files, because everything else
# is read-only.
#

pr_set 69 "Creating Ramdisk"

echo "Creating ramdisk on /tmp"
RAMDISK_SIZE=`get_cfg RAMDISK_SIZE 1024`
/sbin/mke2fs -q -m0 /dev/ram1 ${RAMDISK_SIZE}
/bin/mount -n /dev/ram1 /tmp

pr_set 70 "Setting Hostname"

HOSTNAME=`hostname`
echo "Current hostname: ${HOSTNAME}"
```

```

#####
# Get the IP address of the default server. This is used for
# XDM_SERVER, TELNET_HOST and SYSLOG_HOST if any of them are
# not set explicitly. Default to '192.168.0.254' if it is NOT
# set in the config file.
#
DEFAULT_SERVER=`get_cfg SERVER 192.168.2.100`

>/tmp/mtab

#####
# Load some kernel modules
#

pr_set 71 "Loading Modules"

SERIAL_MOD_LOADED="N"
PARALLEL_MOD_LOADED="N"

KERNEL_VERSION=`cut -d " " -f3 /proc/version`
MODULE_DIR=/lib/modules

for i in 01 02 03 04 05 06 07 08 09 10; do
MODULE=`get_cfg MODULE_${i}`
if [ -n "${MODULE}" ]; then
if [ "${MODULE}" = "serial" ]; then
SERIAL_MOD_LOADED="Y"
fi
if [ "${MODULE}" = "lp" ]; then
PARALLEL_MOD_LOADED="Y"
fi
case "${MODULE}" in

/*) # If it starts with a slash, we use insmod
#
MODULE_PATH="${MODULE_DIR}/${KERNEL_VERSION}${MODULE}"
/sbin/insmod ${MODULE_PATH}
ERR=$?
if [ ${ERR} -ne 0 ]; then
pr_fail
echo
echo "ERROR! loading module: ${MODULE} failed !"
echo
echo -n "Press <enter> to continue "
read CMD
exit
fi
;;

*) # Otherwise, we use modprobe
#
echo "Loading: ${MODULE}"
/sbin/modprobe ${MODULE}

```

```

ERR=$?
if [ ${ERR} -ne 0 ]; then
pr_fail
echo
echo "ERROR! loading module: ${MODULE} failed !"
echo
echo -n "Press <enter> to continue "
read CMD
exit
fi
;;
esac
fi
done

#####
#
# Setup the resolv.conf file
#

pr_set 72 "Setting up resolv.conf"

SEARCH_DOMAIN=`get_cfg SEARCH_DOMAIN`
if [ "${SEARCH_DOMAIN}" != "" ]; then
echo "search ${SEARCH_DOMAIN}" >/tmp/resolv.conf
fi

DNS_SERVER=`get_cfg DNS_SERVER ${DEFAULT_SERVER}`
echo "nameserver ${DNS_SERVER}" >>/tmp/resolv.conf

NFS_SERVER=`get_cfg NFS_SERVER ${DEFAULT_SERVER}`

#####
#
# Setup swap
#
if [ "${USE_NFS_SWAP}" = "Y" ]; then
pr_set 73 "Checking for NFS swap"
modprobe nfsswap
mkdir /tmp/swapfiles
SWAPFILE=/tmp/swapfiles/${HOSTNAME}.swap
SWAP_SERVER=`get_cfg SWAP_SERVER ${NFS_SERVER}`
NFS_SWAPDIR=`get_cfg NFS_SWAPDIR /var/opt/ltsp`
pr_set 74 "Mounting swapfiles directory"
echo "Mounting swapfiles directory"
mount -t nfs -o rsize=2048,wsiz=2048,nolock \
${SWAP_SERVER} : ${NFS_SWAPDIR} /swapfiles \
/tmp/swapfiles
ERR=$?
if [ ${ERR} -ne 0 ]; then
pr_set 74 "Mounting of swap filesystem failed, err=${ERR}"
pr_fail
echo "Mounting of swap filesystem failed, err=${ERR}"
echo "Attempted to NFS mount ${SWAP_SERVER} : ${LTSP_SWAPDIR}"
echo -n "Press <enter> to continue "

```

```

read CMD
exit
else
pr_set 75 "Preparing swapfile"
SWAPFILE_SIZE=`get_cfg SWAPFILE_SIZE 64m`
/sbin/prep_swap -s ${SWAPFILE_SIZE} -f ${SWAPFILE}
ERR=$?
if [ ${ERR} -ne 0 ]; then
pr_set 75 "Error creating swapfile, ERR=${ERR} "
pr_fail
echo "Error creating swapfile, ERR=${ERR} "
echo -n "Press <enter> to continue "
read CMD
exit
fi
pr_set 80 "Formatting Swapfile"
mkswap ${SWAPFILE}
ERR=$?
if [ ${ERR} -ne 0 ]; then
pr_set 80 "Error running mkswapswapfile, ERR=${ERR} "
pr_fail
echo "Error running mkswapswapfile, ERR=${ERR} "
echo -n "Press <enter> to continue "
read CMD
exit
fi
pr_set 83"Enabling swap via NFS"
echo "Enabling swap via NFS"
swapon ${SWAPFILE}
ERR=$?
if [ ${ERR} -ne 0 ]; then
pr_set 83 "swapon failed, ERR=${ERR} "
pr_fail
echo "swapon failed, ERR=${ERR} "
echo
echo "Are you sure the NFS/Swap patch has "
echo "been applied to the workstation kernel?"
echo
echo -n "Press <enter> to continue "
read CMD
exit
fi
fi
fi

#####
#
# Mount filesystems for the FAH client.
# The structure of the mount directory is
# /fah/name_of_workstation Eg /fah/ws001
# The mount point is the same for all workstations:
# /fah
#

if [ "Y" = "Y" ]; then
pr_set 85 "Mounting additional filesystems"
echo "Mounting additional filesystems..."

```

```
mount -t nfs -o nolock ${NFS_SERVER}:/fah/${HOSTNAME} /fah
```

```
fi
```

```
#####
```

```
#
```

```
# Mount filesystems
```

```
#
```

```
#
```

```
# if [ "${LOCAL_APPS}" = "Y" ]; then
```

```
# pr_set 85 "Mounting additional filesystems"
```

```
# echo "Mounting additional filesystems..."
```

```
# mount -t nfs -o nolock ${NFS_SERVER}:/home /home
```

```
# fi
```

```
pr_set 86 "Setting up loopback device"
```

```
echo "Setting up loopback device"
```

```
ifconfig lo 127.0.0.1 netmask 255.0.0.0 broadcast 127.255.255.255
```

```
mkdir /tmp/compiled
```

```
mkdir /tmp/var
```

```
mkdir /tmp/var/run
```

```
mkdir /tmp/var/log
```

```
mkdir /tmp/var/lock
```

```
mkdir /tmp/var/lock/subsys
```

```
mkdir /tmp/var/lib
```

```
mkdir /tmp/var/lib/xkb
```

```
mkdir /tmp/mnt
```

```
#####
```

```
#
```

```
# Check the hostname
```

```
#
```

```
echo "127.0.0.1 localhost ${HOSTNAME}" >/tmp/hosts
```

```
echo "${DEFAULT_SERVER} server" >>/tmp/hosts
```

```
#####
```

```
#
```

```
# Setup the hosts.equiv file
```

```
#
```

```
echo "${DEFAULT_SERVER}" >/tmp/hosts.equiv
```

```
#####
```

```
#
```

```
# Start the syslog daemon
```

```
#
```

```
pr_set 88 "Starting syslogd"
```

```
SYSLOG_HOST=`get_cfg SYSLOG_HOST ${DEFAULT_SERVER}`
```

```
echo "Starting syslogd"
```

```
echo "*.* @${SYSLOG_HOST}" >/tmp/syslog.conf
```

```
syslogd -m 60 -R ${SYSLOG_HOST}
```

```

#####
#
# Local app daemon stuff
#
if [ "${LOCAL_APPS}" = "Y" ]; then
pr_set 90 "Starting Portmapper"
echo "Starting portmapper"
portmap

pr_set 91 "Starting xinetd"
echo "Starting xinetd"
xinetd

NIS_SERVER=`get_cfg NIS_SERVER`
if [ "${NIS_SERVER}" != "" ]; then
pr_set 92 "Setting NIS server"
echo "Setting NIS Server"
echo "ypserver ${NIS_SERVER}" >>/etc/yp.conf
fi

pr_set 93 "Setting domainname"
echo "Setting domainname"
NIS_DOMAIN=`get_cfg NIS_DOMAIN "ltsp"`
echo domainname ${NIS_DOMAIN}
domainname ${NIS_DOMAIN}

pr_set 94 "Starting ypbind"
echo "Starting ypbind"
if [ -z "${NIS_SERVER}" ]; then
ypbind -broadcast
else
ypbind
fi

#
# Need to give ypbind a chance to bind to the server,
# hopefully this is long enough
#
sleep 1

fi

#####
#
# Run the additional rc files.
# These are to make it easier to integrate additional
# functionality into an lts system. Add your scripts to etc/rc.d,
# and put the name of the script in the lts.conf file, and it will
# be executed.
#

pr_set 95 "Checking for rcfiles"
for i in 01 02 03 04 05 06 07 08 09 10; do
RCFILE=`get_cfg RCFILE_${i}`
if [ -n "${RCFILE}" ]; then

```

```
if [ -x /etc/rc.d/${RCFILE} ]; then
/etc/rc.d/${RCFILE}
else
pr_fail
echo
echo " ERROR: RCFILE_${i} is setup in lts.conf, but"
echo " it does not exist in the /etc/rc.d directory"
echo
echo -n "Press <enter> to continue "
read CMD
fi
fi
done

pr_set 100 "Completed"
```

/opt/ltsp/i386/etc/rc.d/startfah

The startfah file you will need to create, it is what each client will run when it boots up. If you will recall this is the same file name we specified in the /opt/ltsp/i386/etc/lts.conf file. The startfah file starts F@H automatically, plain and simple. Note that I added in the *nohup* command, for some reason it would kill off my F@H client when it switched levels, this was a quick and simple fix that solved that problem. However since *nohup* and *nice* aren't part of the standard LTSP files, you will need to copy them from the Farm Server's regular file system and put them the same locations on the /opt/ltsp/i386 file system. They are usually located at:

- /usr/bin/nohup
- /bin/nice

```
#!/bin/bash
# This is the startfah file.

echo "Starting F@H"
cd /fah
nohup ./FAH3Console-Linux.exe > /dev/null &
```

Creating The Extra Directories

Okay, everything we just did was the hard part, now comes the easy part making a few directories and setting up F@H.

First, you will need to make the /fah directory in the /opt/ltsp/i386 file system, you simply run the following command:

```
[root@PE1400SC root]# mkdir /opt/ltsp/i386/fah
```

This is where the folding directory will be mounted, it will look like just /fah on the client side. You don't actually put any files in that directory.

Now you need to create now is all the directories that WILL contain the F@H program & config files.

```
[root@PE1400SC root]# mkdir /fah
[root@PE1400SC root]# cd /fah/
[root@PE1400SC fah]# mkdir ws001
[root@PE1400SC fah]# mkdir ws002
```

Next you need to go into the ws001 directory, and download the [Linux F@H client](#) from the Stanford website. After downloading you run the configuration program and set it up how you want with your username & team and all that. Usually it will start to fold, so break out of the program so you can continue to configure the machine.

You will need to copy the F@H file itself *FAH3Console-Linux.exe* and the *client.cfg* file to the ws002 directory. Do NOT copy any of the other files or the work directory, those are unique to each machine. Don't worry, they will be auto-generated the first time the client runs.

Start Up The Services:

You can use the *chkconfig* program to set the run levels of each of your services, but for now we will just start them manually. Cross your fingers... You need to issue the following commands to startup portmap, bind, tftp, dhcp, nfs:

```
[root@PE1400SC root]# /etc/rc.d/init.d/portmap restart
[root@PE1400SC root]# /etc/rc.d/init.d/named restart
[root@PE1400SC root]# /etc/rc.d/init.d/xinetd restart
[root@PE1400SC root]# /etc/rc.d/init.d/dhcpd restart
[root@PE1400SC root]# /etc/rc.d/init.d/nfs restart
```

If all goes well they should startup just fine, if they give an error you will need to fix whatever is wrong. If you need some help you can post a question in the distributed computing section in [The Overclockers Forum](#).

Setting Up IP Masquerading:

IP Masquerading will essentially forward the packets from your farm network to your home network and out to the Internet, and vice versa. I went to ipmasq.cjb.net to get a [config file](#) to run on the server. The only

thing to check really is to make sure the files are pointing to the proper directories and eth0 & eth1 are set for the proper sides of your network. I then saved this file as /etc/rc.d/rc.firewall on the farm server. Depending on your distro there are different ways to start this file, the easiest for redhat is to add the line to add a line to execute it in your /etc/rc.d/rc.local file.

After setting up the IP Masquerading you can now setup clients and hopefully they will work right!

Setting Up The Clients:

Setting up each client is very easy. The hardest part is you will need to find out the MAC address of the NIC to put in the dhcp.conf file, which I already talked about how to do that earlier.

The Intel NICs & some onboard NICs use PXE to boot, which keeps things nice and simple. I did run into one problem with my Intel NICs not wanting to boot at first, but I discovered it was because they had a really old version of the PXE firmware on them, you can snag a program from Intel that will let you flash the firmware to the latest version.

Since most non-Intel NICs don't have boot ROMs on them, I took the easy way out and created a floppy that has the boot ROM code. I looked around the Internet and to get a burned eprom costs about \$15, however you can get a floppy drive for less, it's your call on which you really want. If you want to purchase the ROMs, a couple places I have found is DisklessWorkstations.com & Linux Central, if anyone knows of other sources please let me know and I can add them to this page. Anyhow, if you go to the ROM-o-matic website, you can generate and download the code to copy to a floppy for your NIC to boot from. I have a mixture of all sorts of NICs, PCI & onboard, and they were all just as easy to setup & boot. The hardest part when adding another client is remembering to add on their configuration data to the different files & creating the folding directory.

So once you have your files configured (and be sure to restart the relevant services), you can boot up your client. I like to have a video card in the system the first time so I can see what is going on. If all goes well you should see it download the proper files over the network and start booting like a regular linux machine. Within a minute or so you should be at a shell prompt on the client machine, which you can then enter the usual set of linux commands to do stuff. I usually just do a quick "ps aux" to see what is running, and you should see the F@H client running somewhere in the list.

Additionally you can check the server's /var/log/messages file for debugging. Another file that I mentioned before is the /fah/ws00x/FAHlog.txt file which outputs what you would usually see on the screen to a file.

Are We Done?!?:

That's pretty much it in a nutshell. It really isn't that complicated if you take it one step at a time. Once you get a single client running, adding more clients is a trivial matter. Be sure to restart the DHCP & NFS server after you edit the configuration files.

You can check out the LTSP site for more info if you would like to run more than just F@H on the workstations, since LTSP was originally created for making terminal servers. You can run X-Windows sessions quite nicely I've heard. Also there is a LTSP-MOSIX project, which adding on MOSIX gives you some nice clustering ability should you need it for special projects.

You don't have to run F@H on your farm, you could just as easily run any of the other many popular distributed projects, as long as they support a Linux client. You could even segment off your farm to have part of them run one project, and another part run another, simply create another script in the /opt/ltsp/i386/etc/rc.d/ directory, and use that different script in the RCFILE line in the /opt/ltsp/i386/etc/lts.conf file.

Questions / Comments?:

Any feedback or questions are welcome, you can [email me directly](#) or you can post a question about it in the distributed computing section on [The Overclockers Forum](#).

Now with all the money you save from not having to buy hard drives or video cards you can purchase more CPUs!

[Back To The Home Page](#)